

**DEVELOPING A LINEAR ALGORITHM FOR
CUBING A CYCLIC PERMUTATION***

Jinyun XUE and David GRIES

Computer Science Department, Cornell University, Ithaca, NY 14853-7501, U.S.A.

Communicated by M. Sintzoff

Received April 1988

Abstract. A linear algorithm is developed for cubing a cyclic permutation stored as a function in an array. This continues work discussed in [1] and [2] on searching for disciplined methods for developing and describing algorithms that deal with complicated data structures such as linked lists. Here, a different representation of a cyclic permutation reveals a simple algorithm; then, an equally simple coordinate transformation is used to yield the final algorithm.

Introduction

A permutation of a finite set of elements is a one-to-one function on the set. For example, viewing a function as a set of ordered pairs, the following is a permutation P on the set $\{a, b, c, d\}$:

$$P = \{(a, b), (b, d), (c, a), (d, c)\}. \quad (1)$$

Thus, $P.a = b$, $P.b = d$, $P.c = a$, and $P.d = c$, where “.” denotes function application.

The product $P * Q$ of two permutations P and Q of the same set is defined by

$$(P * Q).r = P.(Q.r).$$

Further, P^0 is the identity permutation and, for $k \geq 0$, $P^{k+1} = P * P^k = P^k * P$.

Permutation P is *cyclic* if for each r in its domain the set of values $P^i.r$ for $i \geq 0$ is the whole domain. Any permutation can be viewed as the composition of its cyclic components.

Our problem is the following. Given is an array Π containing a permutation P —i.e. $\Pi.r = P.r$ for all r in the domain of P . We desire an algorithm S that cubes Π —i.e. that changes Π to P^3 . Thus, the specification of S is

$$\{\Pi = P\} S \{\Pi = P^3\}. \quad (2)$$

Because cubing a permutation can be done by cubing its cyclic components, we restrict our attention to permutations P that are cyclic.

* This research was supported by the NSF under grant DCR-8320274.

Specification (2) is of little help in the development of S . In order to develop the algorithm, we introduce a second representation of a permutation that does lend insight. We develop an algorithm in terms of this second representation. We then manipulate the algorithm so that a suitable coordinate transformation can be used to yield an algorithm in terms of array Π .

Remark on notation. Sequences are denoted by capital letters and individual elements by small letters. Catenation of sequences and elements is denoted by juxtaposition. For sequence s , $s = s.0 \ s.1 \ \dots \ s.(\#s-1)$, and $s(1..)$ denotes sequence s without its first element.

An algorithm for cubing using a second representation

A cyclic permutation P can be represented by a sequence (consisting of elements of the domain of P) in which the follower of any element r in the sequence is the value $P.r$ (the follower of the last element of the sequence is the first). For example, permutation P given in (1) has the *cyclic representation* $a \ b \ d \ c$.

Second, *any* permutation can be represented by a two-line scheme where the top line is a sequence K giving its domain, the bottom line is a sequence H giving its range, and for each i the pair $(K.i, H.i)$ is in the permutation (see [4]). For example, for permutation P of (1) we have:

$$P = \begin{pmatrix} K \\ H \end{pmatrix} = \begin{pmatrix} a \ b \ c \ d \\ b \ d \ a \ c \end{pmatrix}.$$

We sometimes use the more manageable, linear, notation $P = H : K$. For example, the permutation given above can be written as $(b \ d \ a \ c) : (a \ b \ c \ d)$. Note that there are many two-line representations of P , each derived by reordering the top and bottom sequences in the same manner. For example,

$$P = \begin{pmatrix} a \ b \ c \ d \\ b \ d \ a \ c \end{pmatrix} = \begin{pmatrix} b \ a \ d \ c \\ d \ b \ c \ a \end{pmatrix} = \begin{pmatrix} a \ b \ d \ c \\ b \ d \ c \ a \end{pmatrix}.$$

The rightmost representation has a particularly interesting property: the top sequence $K = a \ b \ d \ c$ is itself a cyclic representation of P . Further, the bottom sequence H is derived from K by rotating K one position to the left. Also, P^2 is $\bar{H} : K$, where \bar{H} is the result of rotating K two positions to the left. This important property is generalized further in the following lemma, whose proof is left to the reader.

Lemma 1. *Let function $rotl.s$ yield sequence s rotated one position to the left. Let K be a cyclic representation of cyclic permutation P . Then, for all i , $i \geq 0$,*

$$P^i = (rotl^i.K) : K.$$

This lemma gives us a simple algorithm for cubing a cyclic permutation $P = H : K$ where K is itself a cyclic representation of P . By the lemma, $H = \text{rotl}.K$, and P^3 can be found simply by rotating H two positions to the left!

An algorithm in terms of H and K

For the moment, suppose the domain of P contains at least three elements. Introducing names for some of the components of P , we write

$$P0: K = x y z W. \quad (3)$$

Lemma 1 tells us that

$$P = H : K \text{ is equivalent to } H = y z W x,$$

$$P^3 = H : K \text{ is equivalent to } H = W x y z,$$

so that an algorithm that cubes P represented by $H : K$ where K is itself a cyclic representation of P can be specified by

$$\{P0 \wedge H = y z W x\} S' \{P0 \wedge H = W x y z\}.$$

A simple way to rotate H is to move the pair y, z past one element of Wx at a time. Introducing two fresh variables U and V , we use the invariant

$$P1: H = U y z V \wedge U V = W x$$

and write the algorithm:

$$\begin{aligned} &U, V := [], Wx; \\ &\text{do } V \neq [] \rightarrow \text{var } v, V' := V.0, V(1..); \\ &\quad H := U v y z V'; \\ &\quad U, V := U v, V' \\ &\text{od.} \end{aligned} \quad (4)$$

Writing an algorithm in terms of Π

Given $P = H : K$, the above algorithm changes H to establish $P^3 = H : K$. Our desired algorithm is to be in terms of array Π , where Π , H , and K are coupled by the following *coupling invariant*:

$$I: \Pi = H : K. \quad (5)$$

We use invariant I to help rewrite algorithm (4) in terms of Π . Note that an iteration of the loop of (4) changes H from

$$U y z v V' \text{ to } U v y z V'.$$

This requires a change in the elements of Π , but which elements? To answer this question, we introduce fresh variables p, q, r to denote elements and fresh variables \bar{U}, \bar{V} to denote sequences and define them by

$$P2: K = \bar{U} p q \bar{V} \wedge \# U = \# \bar{U} \wedge r = (\bar{V} \bar{U} p).0.$$

Then, when H is changed as above, the permutation

$$\begin{pmatrix} \bar{U} p q r \bar{V}' \\ U y z v V' \end{pmatrix} \text{ is changed to } \begin{pmatrix} \bar{U} p q r \bar{V}' \\ U v y z V' \end{pmatrix} \quad (\text{for } \bar{V}' \text{ defined by } \bar{V} = r \bar{V}').$$

This means that only the values $\Pi.p$, $\Pi.q$, and $\Pi.r$ have to be changed—to v , y , and z , respectively.

Let us first change algorithm (4) to reflect the introduction of the new variables defined by $P2$. Note that whenever an element is appended to U a change in \bar{U} , p , q , r , and \bar{V} is required, because the lengths of U and \bar{U} are the same (by $P2$).

$$\begin{aligned} &U, V := [], Wx; \\ &p, q, r, \bar{U}, \bar{V} := x, y, z, [], W; \\ &\text{do } V \neq [] \rightarrow \text{var } v, V' := V.0, V(1..); \\ &\quad H := U v y z V'; \\ &\quad U, V := Uv, V'; \\ &\quad \bar{U}, p, q, r, \bar{V} := \bar{U} p, q, r, (\bar{V}' \bar{U}' p).0, \bar{V}'(1..) \\ &\text{od.} \end{aligned} \tag{6}$$

Making the coordinate transformation

We are now ready to replace references to $U, V, H, W, V', \bar{U}, \bar{V}, \bar{V}', y$, and z by references to Π using coupling invariant I as well as the invariants $P0$, $P1$, and $P2$. Only the variables Π, p, q, r , and an arbitrary element x of the domain of P remain.

First, consider the initialization. Initially, from the fact that K is a cyclic representation of permutation P and from I we have $y = \Pi.x$ and $z = \Pi.y$. Hence, the initialization becomes

$$p := x; \quad q := \Pi.x; \quad r := \Pi.q.$$

The following derivation shows that the expression $V \neq []$ is equivalent to $r \neq x$:

$$\begin{aligned} V \neq [] &\equiv \bar{V} \neq [] && (\text{by } P1 \text{ and } P2) \\ &\equiv r = (\bar{U} p).0 && (\text{by } P2) \\ &\equiv r = x && (\text{by } P0 \text{ and } P2). \end{aligned}$$

The statement $H := U v y z V'$ calls for a change in Π . Π is being changed from

$$\Pi = \begin{pmatrix} \bar{U} p q r \bar{V} \\ U y z v \bar{V}' \end{pmatrix} \text{ to } \Pi = \begin{pmatrix} \bar{U} p q r \bar{V} \\ U v y z \bar{V}' \end{pmatrix} \quad \text{where } \# U = \# \bar{U}.$$

Hence, $\Pi.p$, $\Pi.q$, and $\Pi.r$ are the only values of Π that need to be changed, and the change is effected by the assignment

$$\Pi.p, \Pi.q, \Pi.r := \Pi.r, \Pi.p, \Pi.q.$$

Finally, in the assignment to p , q , and r , we need to replace the reference to $(\bar{V}\bar{U}p).0$, by a reference to $\Pi.r$.

The final algorithm is

$$\begin{aligned} & p := x, \quad q := \Pi.p; \quad r := \Pi.q; \\ & \text{do } r \neq x \rightarrow \Pi.p, \Pi.q, \Pi.r := \Pi.r, \Pi.p, \Pi.q; \\ & \quad p, q, r := q, r, \Pi.p \\ & \text{od.} \end{aligned} \tag{7}$$

Let us now consider the cases where the domain of P has size one or two. In both cases, $P = P^3$. In both cases, execution of algorithm (7) stores the same value in p and x and then terminates after 0 iterations of the loop, so Π remains unchanged and contains P^3 .

Remark. The extension to compute P^k for some $k \geq 0$ should be obvious.

Acknowledgment

Thanks go to Rett Bull for comments on earlier drafts of this report.

References

- [1] W.H.J. Feijen, A.J.M. van Gasteren and D. Gries, In-situ inversion of a cyclic permutation, *Info. Proc. Lett.* **24** (1987) 11–14.
- [2] D. Gries and J.F. Prins, McLaren's masterpiece, *Sci. Computer Programming* **8** (1987) 139–145.
- [3] D. Gries, *The Science of Programming* (Springer, New York, 1981) 265–274.
- [4] D.E. Knuth, *The Art of Computer Programming*, Vol. 1 (Addison-Wesley, Menlo Park, 1973).